What is claimed is :

CLAIMS

1) A method of detecting congestion, comprising:
setting at least one threshold value for buffer occupancy; and
dropping all packets when an average queue size exceeds said threshold.

2) The method according to claim 1, wherein said threshold comprises a minimum and a maximum threshold.

3) The method according to claim 2, further comprising the steps of:
marking packets when said average queue size is between said minimum threshold and said maximum threshold; and
dropping all packets when an average queue size equals or exceeds said maximum threshold.

4) The method according to claim 1, wherein said threshold comprises a minimum threshold, a maximum threshold and a feedback threshold.

5) The method according to claim 4, further comprising the steps of:
marking outgoing packets when said average queue size is between said minimum threshold and said feedback threshold;
dropping incoming packets and marking said outgoing packets when said average queue size is between said feedback threshold and said maximum threshold; and
dropping all of said incoming packets when an average queue size equals or exceeds said maximum threshold.

6) The method according to claim 5, wherein said average queue size is a size of a bucket of a token bucket filter.

7) The method according to claim 5, wherein said packets are marked deterministically in said step of marking outgoing packets when said average queue size is between said minimum threshold and said feedback threshold.

8) The method according to claim 5, wherein said packets are dropped probabilistically in said step of dropping incoming packets when said average queue size is between said feedback threshold and said maximum threshold.

9) The method according to claim 5, wherein said step of marking outgoing packets when said average queue size is between said minimum threshold and said feedback threshold comprises setting at least one bit in at least one of said packets, and said step of marking said outgoing packets when said average queue size is between said feedback threshold and said maximum threshold comprises setting at least one bit in at least one of said packets.

10) The method according to claim 5, further comprising the step of calculating said average queue size based on a moving average.

11) The method of according to claim 5, further comprising the steps of:
varying a number of tokens consumed by a data packet; and
transmitting said packet if the number of tokens consumed by said packet is less than or equal to available tokens.

12) The method according to claim 9, wherein said bit is part of a type of services byte.

13) The method according to claim 9, wherein a core node performs said step of setting said bit.

14) The method according to claim 9, wherein said bit is an LCN bit.

15) The method according to claim 10, wherein said moving average is an exponential moving average.

16) The method according to claim 11, wherein said step of varying the number of tokens consumed by a data packet comprises varying the number of tokens consumed by data of unit size.

17) The method according to claim 11, wherein said step of varying said number of tokens comprises the step of decreasing data of unit size monotonically if demand increases monotonically during a congestion free period.

18) The method according to claim 11, wherein said step of varying said number of tokens comprises the step of increasing data of unit size upon receipt of a message.

19) The method according to claim 12, wherein said type of services byte is a differentiated services code point byte.

20) The method according to claim 15, wherein said packets are marked deterministically in said step of marking outgoing packets when said average queue size is between said minimum threshold and said feedback threshold, wherein said packets are dropped probabilistically in said step of dropping incoming packets when said average queue size is between said feedback threshold and said maximum threshold, and wherein said step of marking outgoing packets when said average queue size is between said minimum threshold and said feedback threshold comprises a core node setting at least one bit in at least one of said packets and said step of marking outgoing packets when said average queue size is between said feedback threshold and said maximum threshold comprises setting at least one bit in at least one of said packets, wherein said bit is part of a type of services byte.

21) The method according to claim 18, wherein said step of varying said number of tokens comprises further decreasing said data of unit size if an average queue size is greater than a demand threshold.

22) A method of regulating traffic flow between nodes, comprising:
detecting congestion;
sending a message to at least one node;
regulating at least one traffic rate of said at least one node; and
detecting when congestion is clear.

23) The method according to claim 22, further comprising the step of incrementing said at least one traffic rate at random times when said congestion is clear.

24) The method according to claim 22, wherein a core node detects said congestion, and an output node sends said message to at least one input node.

25) The method of according to claim 22, wherein said step of regulating at least one traffic rate of said at least one node, comprises:
reducing said at least one traffic rate of said at least one node proportional to the amount of traffic that said at least one node is injecting when said congestion is detected.

26) The method of according to claim 22, wherein said step of regulating at least one traffic rate of said at least one node, comprises:
varying a number of tokens consumed by a data packet; and
transmitting said packet if a number of tokens consumed by said packet is less than or equal to available tokens.

27) The method according to claim 22, wherein said step of sending a message to at least one node comprises marking at least one packet.

28) The method of according to claim 22, wherein said step of regulating at least one traffic rate of said at least one node, comprises steps of:
varying a number of tokens consumed by a data packet by varying the number of tokens consumed by data of unit size, comprising the steps of :

decreasing said data of unit size monotonically if demand increases monotonically during a congestion free period;

further decreasing said data of unit size if an average queue size is greater than a demand threshold; and

increasing said data of unit size upon receipt of a message; and

transmitting said packet if a number of tokens consumed by said packet is less than or equal to available tokens, wherein said at least one packet is marked deterministically.

29) The method according to claim 26, wherein said step of varying the number of tokens consumed by a data packet comprises varying the number of tokens consumed by data of unit size.

30) The method according to claim 26, wherein said step of varying said number of tokens comprises decreasing data of unit size monotonically if demand increases monotonically during a congestion free period.

31) The method according to claim 26, wherein said step of varying said number of tokens comprises increasing data of unit size upon receipt of a message.

32) The method according to claim 27, wherein said packet is marked deterministically.

33) The method according to claim 27, wherein said step of marking packets comprises setting at least one bit in at least one of said packets.

34) The method according to claim 30, wherein said step of varying said number of tokens comprises further decreasing said data of unit size if an average queue size is greater than a demand threshold.

35) The method according to claim 33, wherein said bit is part of a type of services byte.

36) A method of controlling traffic flow in a differentiated services domain, comprising:
varying a number of tokens consumed by a data packet; and
transmitting said data packet if a number of tokens consumed by said data packet is less than or equal to available tokens.

37) The method according to claim 36, wherein said step of varying the number of tokens consumed by a data packet comprises varying the number of tokens consumed by data of unit size.

38) The method according to claim 36, wherein said number of tokens consumed by a data packet is varied based on state and demand.

39) The method according to claim 36, further comprising steps of:
a first step of determining available bandwidth by calculating an average queue size at a token bucket filter; and
a second step of determining if said average queue size is greater than a demand threshold.

40) The method according to claim 36, wherein said step of varying said number of tokens consumed by a data packet comprises a step of decreasing data of unit size monotonically if demand increases monotonically during a congestion free period.

41) The method according to claim 36, wherein said step of varying said number of tokens comprises a step of increasing data of unit size upon receipt of a message.

42) The method according to claim 37, wherein said step of varying the number of tokens consumed by data of unit size comprises varying the number of tokens consumed by a byte of data.

43) The method according to claim 37, wherein said step of varying the number of tokens consumed by data of unit size, comprises varying the number of tokens consumed by a bit of data.

44) The method according to claim 37, wherein a minimum number of tokens consumed by said data of unit size equals a depth of a token bucket filter divided by a min-max fair share of the buffer.

45) The method according to claim 37, wherein a minimum number of tokens consumed by said data of unit size equals a depth of a token generation rate divided by a min-max fair share of the bandwidth.

46) The method according to claim 37, wherein a maximum number of tokens consumed by said data of unit size equals a depth of a token bucket filter divided by a maximum transmission unit.

47) The method according to claim 37, wherein a maximum number of tokens consumed by said data of unit size equals a token generation rate divided by a minimum data buffer drain rate.

48) The method of according to claim 37, wherein said step of varying the number of tokens consumed by data of unit size, comprises the steps of :

decreasing said data of unit size monotonically if demand increases monotonically during a congestion free period;

further decreasing said data of unit size if an average queue size is greater than a demand threshold; and

increasing said data of unit size upon receipt of a message, wherein said number of tokens consumed by a data packet is varied based on state and demand.

49) The method according to claim 38, wherein said state is a congestion state.

50) The method according to claim 40, wherein said step of varying said number of tokens comprises further decreasing said data of unit size if an average queue size is greater than a demand threshold.

51) An apparatus for controlling traffic flow, comprising:
a plurality of nodes; and
at least one token bucket filter corresponding to at least one of said nodes.

52) The apparatus according to claim 51, wherein said token bucket filter comprises:
a token generator; and
a bucket operably connected to said token generator to hold at least one generated token.

53) The apparatus according to claim 52, wherein said nodes comprises:
at least one input node;
at least one output node; and
at least one core node.

54) The apparatus according to claim 53, wherein said at least one core node does not maintain per-flow state information and carry a large number of aggregated flows; and wherein said output node and input node maintain per-flow state.

55) The apparatus according to claim 53, wherein said nodes are part of a domain.

56) The apparatus according to claim 53, wherein said nodes are part of an Internet.

57) The apparatus according to claim 55, wherein said domain is a differentiated services domain.